AMENDMENTS TO THE CLAIMS

1. (Original) A network router system comprising a switch fabric configured such that information flows through said switch fabric from a plurality of source ports to a plurality of destination ports in substantially fixed sized chunks, each said chunk being formatted to include a framing symbol.

2. (Original) The router system of claim 1 wherein said framing symbol is located adjacent the trailing end of each said chunk

3. (Original) The router system of claim 1 wherein said framing symbol has a length of 16 bytes.

a3

4. (Original) The router system of claim 3 wherein said framing symbol includes two bits of said framing symbol intermixed in each of 56 contiguous bytes of said chunk immediately followed by two contiguous bytes of said framing symbol.

5. (Original) The router system of claim 1 wherein each said chunk is further formatted to include forward error correction (FEC) coding.

6. (Original) The router system of claim 4 wherein said FEC coding is located adjacent and following said framing symbol.

7. (Original) The router system of claim 1 wherein each said chunk is further formatted to include a preamble, said preamble containing information configured to allow alignment of router clock and data recovery circuitry.

8. (Original) The router system of claim 7 wherein each said chunk is further formatted to include a "Break Bytes" field and a "Make Bytes" field, said fields configured to precondition an optical receiver prior to the arrival of a chunk.

9. (Original) The router system of claim 8 wherein said "Break Bytes" field is located ahead of said "Make Bytes" field in a chunk.

10. (Original) The router system of claim 9 wherein said "Make Bytes" field is located ahead of said preamble.

11. (Original) The router system of claim 4 wherein each said chunk is further formatted to include a chunk cyclical redundancy check (CRC) field.

12. (Original) The router system of claim 11 wherein said chunk CRC field is located adjacent and preceding said two contiguous bytes of said framing symbol.

13. (Original) The router system of claim 1 wherein each said chunk is further formatted to include a scrambler seed.

14. (Original) The router system of claim 1 wherein each said chunk is further formatted to include a chunk header.

15. (Original) The router system of claim 14 wherein said chunk header includes identification of chunk type.

a3

16. (Original) The router system of claim 14 wherein said switch fabric is partitioned into a plurality of working subplanes.

17. (Original) The router system of claim 16 wherein said chunk header includes identification of a specific routing subplane through said switch fabric.

18. (Original) The router system of claim 14 wherein said chunk header includes a header parity.

19. (Original) The router system of claim 14 wherein said chunk header includes identification of source port and destination port for said chunk.

20. (Original) The router system of claim 14 wherein said chunk header includes a master chunk bit.

21. (Original) The router system of claim 14 wherein said chunk header includes a sequence number.

22. (Currently Amended) The router system of claim 1 wherein each said chunk further contains a chunk ~~data~~ payload comprising at least one packet segment and an associated packet header.

23. (Currently Amended) The router system of claim 22 wherein said at least one packet segment is selected from the group consisting of ~~a segment of an IP~~ portions of packets ~~that is~~ that are larger than said chunk ~~data~~ payload ~~and multiple complete IP, packet that are~~ portions of packets that are equal in size to said chunk payload, and portions of packets that are smaller in size ~~each not larger~~ than said chunk ~~data~~ payload.

24. (Original) The router system of claim 1 wherein said switch fabric comprises an optical switch.

25. (Original) A method of information flow through a network router system comprising:

encapsulating input data packets from a plurality of source ports into substantially fixed sized chunks;

formatting overhead information onto each of said chunks, said overhead including a framing symbol;

a3

directing said chunks through a switch fabric toward a plurality of destination ports;

performing error detection and error correction on said chunks;

removing said overhead information from said chunks; and

reassembling said input data packets out of said chunks.

26. (Original) The method of claim 25 wherein all information flows through said switch fabric in said substantially fixed sized chunks.

27. (Original) The method of claim 25 wherein said framing symbol is located adjacent the trailing end of each said chunk.

28. (Original) The method of claim 25 wherein said framing symbol has a length of 16 bytes.

29. (Original) The method of claim 28 wherein said framing symbol includes two bits of said framing symbol intermixed in each of 56 contiguous bytes of said chunk immediately followed by two contiguous bytes of said framing symbol.

30. (Original) The method of claim 25 wherein said formatting includes forward error correction (FEC) coding in each chunk.

31. (Original) The method of claim 30 wherein said formatting includes cyclical redundancy check (CRC) coding in each chunk.

32. (Original) The method of claim 25 wherein said formatting includes adding a scrambler seed in each chunk.

33. (Original) The method of claim 25 wherein said formatting includes adding a chunk preamble in each chunk.

34. (Original) The method of claim 33 wherein said chunk preamble is programmable in length.

35. (Original) The method of claim 25 wherein said formatting includes a "Break Bytes" field and a "Make Bytes" field, said fields configured to precondition an optical receiver prior to the arrival of a chunk.

a3

36. (Original) The method of claim 35 wherein said "Break Bytes" field and said "Make Bytes" field are programmable in length.

37. (Original) The method of claim 25 wherein said formatting includes adding a chunk header in each chunk.

38. (Original) The method of claim 37 wherein said chunk header includes identification of chunk type.

39. (Original) The method of claim 37 wherein said switch fabric is partitioned into a plurality of working subplanes.

40. (Original) The method of claim 39 wherein said chunk header includes identification of a specific routing subplane through said switch fabric.

41. (Original) The method of claim 37 wherein said chunk header includes a header parity.

42. (Original) The method of claim 37 wherein said chunk header includes identification of source port and destination port for said chunk.

43. (Original) The method of claim 37 wherein said chunk header includes a master chunk bit.

44. (Original) The method of claim 37 wherein said chunk header includes a sequence number.

45. (Original) The method of claim 25 wherein said switch fabric incorporates an optical switch.

46. (Original) The method of claim 38 wherein said directing comprises using said identification of chunk type in said chunk header to enable guaranteed bandwidth chunks to pass ahead of best effort chunks through said switch fabric.

a3

47. (Original) The method of claim 40 wherein said directing comprises using said identification in said chunk header of a specific routing subplane to route said chunks through said switch fabric.

48. (Original) The method of claim 42 wherein said directing comprises using said identification in said chunk header of source port and destination port to route said chunks from a selected source port to a selected destination port.

49. (Original) The method of claim 33 wherein said directing comprises using said chunk preamble to phase align circuitry to retrieve data from said chunk.

50. (Original) The method of claim 35 wherein said directing comprises using said "Break Bytes" field and said "Make Bytes" field to precondition an optical receiver prior to the arrival of a chunk.

51. (Original) The method of claim 50 wherein said "Break Bytes" field maintains a 50 percent density of ones and zeros for a laser beam.

52. (Original) The method of claim 50 wherein said "Make Bytes" field reestablishes a decision threshold level of a limiting amplifier within a burst mode optical receiver.

53. (Original) The method of claim 27 wherein said directing comprises using said framing symbol in each said chunk to determine uniquely within a stream of bits the beginning and the trailing end of said chunk.

54. (Original) The method of claim 32 wherein said formatting comprises using said scrambler seed in said chunk to balance zeroes and ones and to minimize run lengths of zeroes and ones by scrambling bits across said chunk.

55. (Original) The method of claim 31 wherein said performing error detection and correction comprises using said FEC encoded in each said chunk to detect and correct errors in said chunk.

a3

56. (Original) The method of claim 55 wherein said performing error detection and correction comprises using said CRC encoded in each said chunk to determine that the entire said chunk has a proper CRC value after said FEC error detection and correction.

57. (Original) The method of claim 44 wherein said performing error detection and correction comprises using said sequence number in said chunk header for alarming and for alerting that a chunk potentially was corrupted.

58. (Original) The method of claim 57 wherein a re-initialize bit is used to enable reinitialization of said sequence number, such that said alarming is avoided.

59. (Original) The method of claim 42 wherein said performing error detection and correction comprises using said identification in said chunk header of source port and destination port to verify the route of said chunks from a selected source port to a selected destination port.

60. (Original) The method of claim 25 wherein said substantially fixed sized chunks each have a length of approximately 400 bytes.

61. (Original) The method of claim 60 wherein said fixed sized chunk contains multiple data packets.

62. (Original) The method of claim 60 wherein said fixed sized chunk contains a segment of a data packet, said data packet having a length greater than 400 bytes.